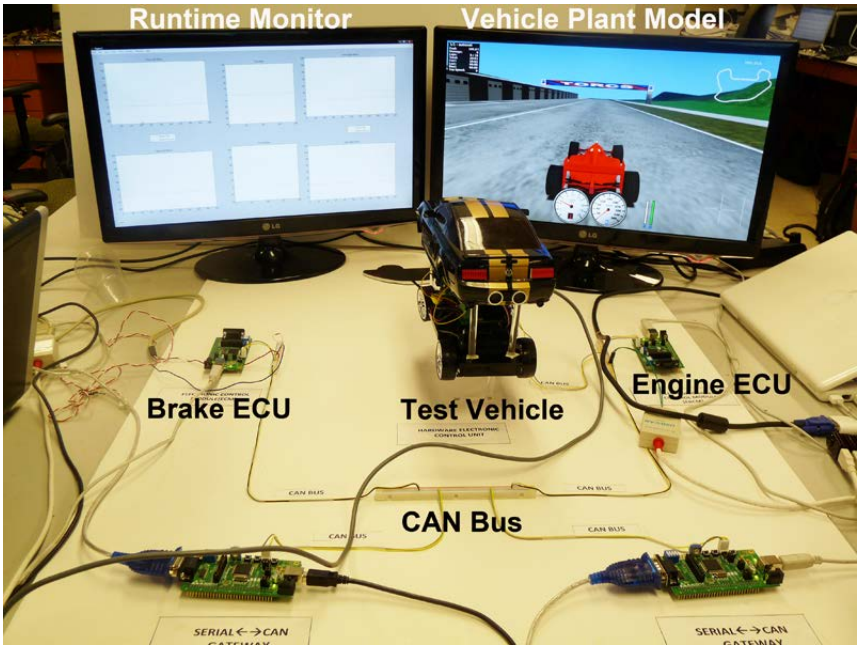
AUTOPLUG:
Remote Diagnostics Automotive Architecture for Control Software Safety

Rahul Mangharam, Yash V. Pant and Truong X. Nghiem
Department of Electrical & Systems Engineering
University of Pennsylvania



The Problem:

In 2010, over 20.3 million vehicles were recalled. Software issues related to automotive controls such as cruise control, anti-lock braking system, traction control and stability control, account for an increasingly large percentage of the overall vehicles recalled. There is a need for new and scalable methods to evaluate automotive controls in a realistic and open setting. We have developed AutoPlug, an automotive Electronic Controller Unit (ECU) test-bed to diagnose, test, update and verify controls software. AutoPlug consists of multiple ECUs interconnected by a CAN bus, a racecar driving simulator which behaves as the plant model and a vehicle controls monitor in Matlab. As the ECUs drive the simulated vehicle, the physics-based simulation provides feedback to the controllers in terms of acceleration, yaw, friction and vehicle stability. This closed-loop platform is then used to evaluate multiple vehicle control software modules such as traction, stability and cruise control. With this test-bed we are aim to develop ECU software diagnosis and testing to evaluate the effect on the stability and performance of the vehicle. Code updates can be executed via a smart phone so drivers may remotely "patch" their vehicle. This closed-loop automotive control test-bed allows the automotive research community to explore the capabilities and challenges of safe and secure remote code updates for vehicle recalls management.

We have demonstrated AutoPlug to the U.S. Department of Transportation, Research and Innovative Technology Administration, John A. Volpe National Transportation Systems Center. We have also demonstrated this to the National Academy of Engineers (NAE), Intel Corporation, Toyota InfoTech Center, BOSCH Research and General Motors.

Keywords
Automotive safety recalls, Control Systems Diagnostics, Stability Control, Traction Control, Anti-lock Braking, Adaptive Cruise Control

Software-related Automotive Recalls

The increasing complexity of software in automotive systems has resulted in the recent rise of firmware-related vehicle recalls due to undetected bugs and software faults. In 2009, Volvo recalled 17,614 vehicles due to a software error in the engine cooling fan control module[1]. According to the NHSTA report, the error could result in engine failure and could possibly lead to a crash. In August 2011, Jaguar recalled 17,678 vehicles due to concerns that the Cruise Control in those vehicles may not respond to normal inputs and once engaged could not be switched off[2]. In November 2011, Honda recalled 2.5 million vehicles to update the software that controls their automatic transmissions[3].

While there is a significant effort for automotive software testing and verification at the design stage (e.g. AUTOSAR), not all possible runtime faults throughout the vehicle's lifetime can be detected. A systematic approach and infrastructure for post-market runtime diagnostics for the control software is lacking in current automotive systems. Once the vehicle leaves the dealership lot, its performance and operation safety is a black box to the manufacturers and the original equipment providers. For the over 100 million lines of code and over 60 Electronic Control Units (ECUs) in a modern vehicle, there are only about 8 standard Diagnostic Trouble Codes (DTCs) for software, and those too are generic (e.g. memory corruption). Out of the DTCs for software, none target the *control software* in the ECUs even though control systems like stability control, cruise control, and traction control are safety-critical systems.

Approach: Runtime in-vehicle Diagnostics and Recalls Management

The current approach for handling vehicle recalls is reactive where the manufacturers announce a recall only after the problem occurs in a significant population of deployed vehicles and all vehicles of that particular year/make/model are recalled. A software recall involves the vehicle being taken to service center and a technician either manually replaces the ECU which contains the faulty code, or reprograms the code onboard the ECU with the new version provided by the manufacturer. One problem with this method is that the decision to recall vehicles involves word-of-mouth or manually logged information going from the service centers to the manufacturer, which takes time and in the meanwhile may result in a malfunction within a safety critical system. This wait-and-see approach to recalls has a significant cost in both time and money and has a negative impact on the vehicle manufacturers reputation.

The above observations lead us to believe that there is a need for systematic post-market in-vehicle diagnostics for control systems software such that issues can be detected early. The in-vehicle system would be responsible for data logging of sensor values and runtime evaluation of controller states. To complement this, a Remote Diagnostics Center (RDC) would receive this data, over a network link, to prepare an appropriate Fault Detection and Isolation (FDI) response. This would normally be in the form of sending a custom Dynamic Diagnostic Code that observes the ECUs and controller tasks in question. Once sufficient data is captured, the RDC, using a model of the plant, is able to execute a grey-box structured system identification to build a plant model of the particular vehicle. Using this vehicle-specific plant model, the RDC develops a fault-tolerant controller for the issue and the vehicle is remotely re-programmed via a code update. While this approach is difficult in practice, as it would require extensive runtime verification of the patched controller, we present the early design of such a system with AutoPlug.

We have developed AutoPlug, an automotive Electronic Controller Unit (ECU) test-bed to diagnose, test, update and verify controls software. AutoPlug consists of multiple ECUs interconnected by a CAN bus, a vehicle driving simulator which behaves as the plant model and a vehicle controls monitor for system identification, diagnostics and remote programming of ECUs.

---

[1] NHTSA Campaign ID: 09V218000. www.safecar.gov

[2] Jaguar Software Issue May Cause Cruise Control to Stay On. http://spectrum.ieee.org

[3] Honda recalls 2.5 million vehicles on software issue. http://www.reuters.com

AutoPlug is focused on addressing software-related recalls for vehicles.

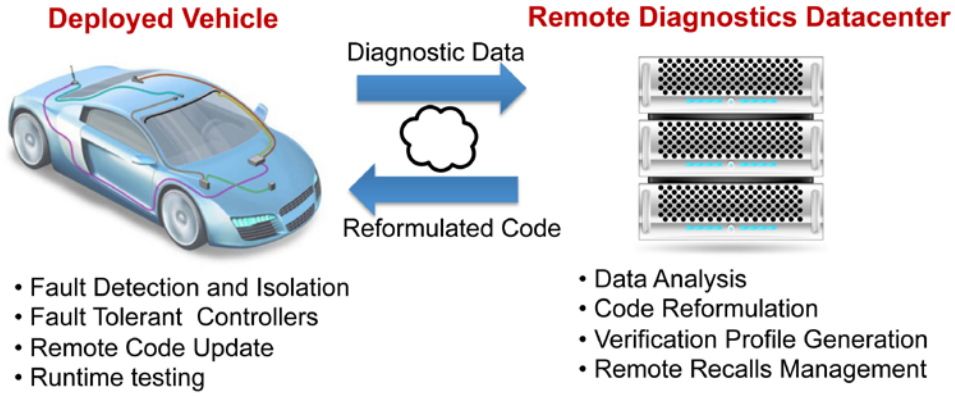


Figure 1. Overview of AutoPlug services

Methodology

The AutoPlug automotive architecture aims to make the vehicle recalls process a less reactive one with a runtime system for diagnosis of automotive control systems and software. Our focus is on the on-line analysis of the control system and control software within the vehicle ECU network and between the vehicle and the RDC. We assume the network link between the two is available. The runtime system *within the vehicle* is responsible for:

1. *Fault Detection and Isolation (FDI):* Sensor, actuator and controller states are logged for the specific ECU. This data is analyzed locally and a summary of the states are transmitted to the RDC.
2. *Fault Tolerant Controllers:* Once a fault is detected, the high-performance controller is automatically replaced with a backup controller.
3. *ECU re-programming for remote code updates:* Upon reception of updated controller code from the RDC, the runtime system re-flashes the particular controller task(s) with the updated code.
4. *Patched Controller runtime-verification:* The updated code is monitored with continuous checks for safety and performance. We do not focus on this aspect in this paper but consider it in future work.

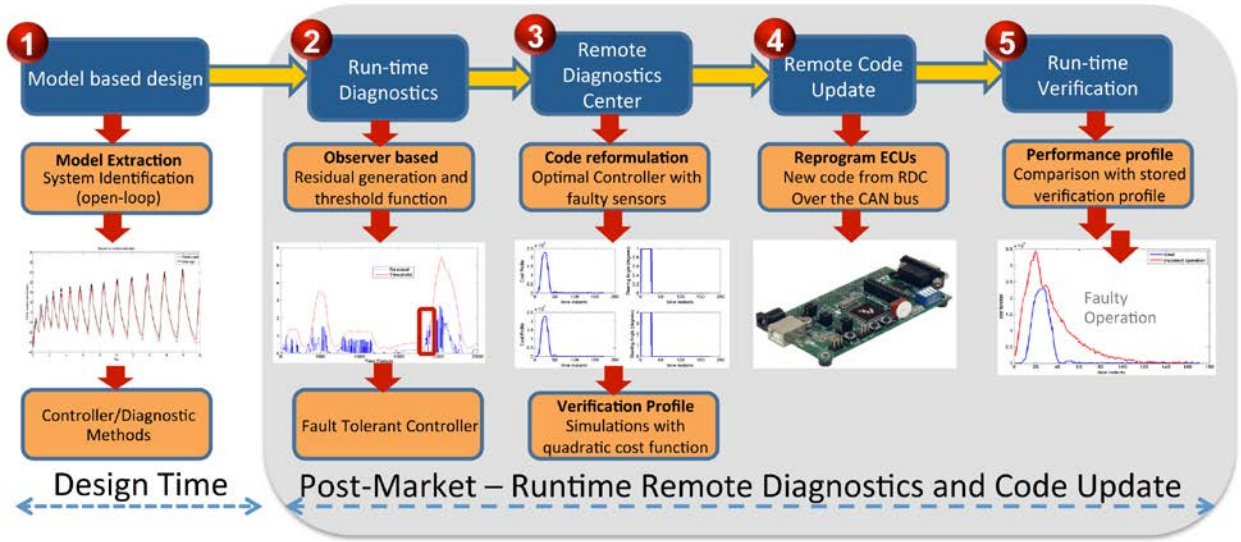


Figure 2. Stages of the AutoPlug architecture

While the on-board system provides state updates of the specific controller, the *Remote Diagnostics Center (RDC)* provides complementary support by:

1. *Data analysis and fault localization:* Using grey-box structured system identification, a plant model of the particular vehicle is created. The existing controller is evaluated on this model to isolate faulty behavior.
2. *Reformulating Control and Diagnostics Code:* A new controller is formulated for the specific plant model and further diagnostics code is dispatched.
3. *Recalls Management:* Reformulated controller code is transmitted to the vehicle.
4. *Generating Controller Verification profiles:* The updated controller is probed for performance and safety.

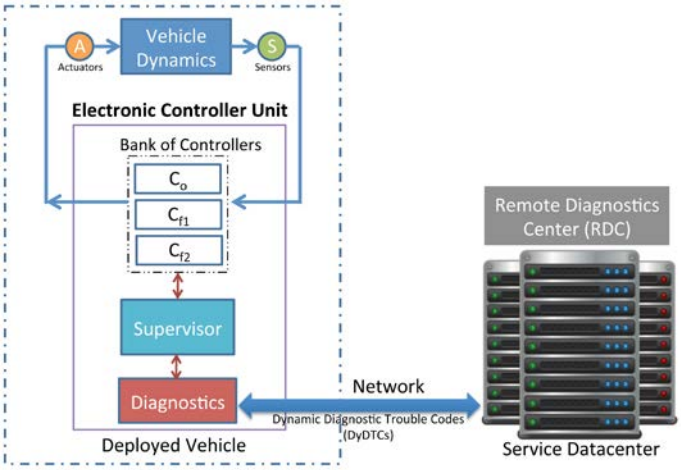


Figure 3. AutoPlug Remote Diagnostic Center interfacing with vehicle controllers

A more descriptive view is provided in Fig 1. Further explanation of these services is presented in our technical report[4].

Findings

A. AutoPlug Architecture

Regular firmware updates over a network for infotainment systems in vehicles is now a rising trend. A central node keeps track of the firmware version for a large number of cars and when a new version is ready to be launched, updates the firmware code on each vehicle based on its difference with the existing version on the vehicle. Similar methods for the control systems and diagnostics code implemented on board the ECUs of vehicles have yet to be developed and implemented. This is because the safety-critical nature of the control systems implemented on the ECUs makes the task of coming up with these methods and proving their correctness more exacting.

The working of the AutoPlug scheme can be divided into five steps, as shown in Fig. 2. At the design stage, a model for the automotive system is extracted and is used for the controller formulation and the model-based diagnostic methods. After the vehicle has been deployed, run-time diagnostics onboard the vehicle detect and isolate faults, and the knowledge of that is used to switch to a fault tolerant controller. The diagnostic data logged in the vehicle is then sent to the RDC where the new controller/diagnostics code is formulated. The reformulated code, along with a verification profile for run-time evaluation is sent to the vehicle and reprogrammed onto the ECUs.

This section will explain the key parts of the AutoPlug architecture shown in Fig. 3 and provide an overview of their functions.

---

[4] Yash V. Pant, Miroslav Pajic and Rahul Mangharam. AUTOPLUG: Architecture for Remote Electronic Controller Unit Diagnostics in Automotive Systems. University of Pennsylvania - Technical Report. http://repository.upenn.edu/mlab_papers/49/

1. Vehicle Dynamics and Sensors/Actuators

Measurements, like yaw rate and roll angle, are available as messages on the CAN bus through the corresponding sensors during vehicle operation. Actuators (e.g. throttle, brakes) can also be commanded by corresponding CAN messages. In addition, an approximation of the vehicle dynamics (e.g. powertrain dynamics, lateral dynamics) is available at the design stage. This knowledge of the vehicle dynamics and the run-time sensor measurements from the vehicle and actuation are used for the control and diagnostics onboard the vehicle.

2. Bank of Controllers

Onboard the ECUs, multiple controllers are implemented to achieve the same performance criteria while using different subsets of available sensors. Normal operation involves the nominal controller using all the available (and needed) sensors for feedback control, and the sensor status is monitored by the diagnostics. Based on the status of sensors, only one of the controllers is active at a time. This bank of controllers aims to achieve the predefined control objective despite a sensor or set of sensors providing faulty measurements. Each individual controller corresponds to one case of sensor fault(s) and is activated whenever that fault or set of faults is detected. These controllers through either *Accommodation* or *Reconfiguration* may meet the control objective. Based on the information from the *Onboard Diagnostics*, the *Supervisor Logic* takes the original controller (for the no fault case) out of the loop and switches to the corresponding fault tolerant controller in run-time.

3. On-board Diagnostics

Run-Time FDI is one of the principal components of the AutoPlug architecture. The onboard diagnostics code can be on a single ECU or distributed among different ECUs. The scheme which forms the backbone of the onboard diagnostics is tasked with monitoring sensor measurements and detecting any sensor faults and isolating the fault(s) to one or more sensors. A detailed explanation of the FDI scheme employed is outlined[4].

4. Supervisor

The supervisor in Fig. 2 is implemented onboard an ECU to overlook the functioning of the fault-tolerant architecture. The supervisor receives information from the onboard diagnostics module and switches between the original (no fault) controller and the fault-tolerant controllers based on which sensor or set of sensors is faulty. The supervisor is also responsible for logging run-time sensor measurements, control inputs, and diagnostics information for a finite window of time. This data logging continues for some time after the fault is detected, and is then transmitted to the Remote Diagnostics Center. In addition, after the remote controller code update, the supervisor is responsible for monitoring the performance of the new controller.

5. The Remote Diagnostics Center (RDC)

The RDC is a facility of the vehicle's manufacturer that is connected to all deployed vehicles and provides remote recalls management. The RDC performs tasks that may be autonomous or have a human in the loop. In the scope of this paper, the RDC performs autonomous tasks like controller reformulation and generation of a verification profile for patched controller evaluation onboard the vehicle at run-time. The RDC receives data from the vehicle whenever the vehicle encounters a fault. The logged data from the vehicle is used for the diagnostic tasks performed at the RDC and the reformulated controller is sent to the vehicle to be reprogrammed onto the ECUs.

After a fault has been detected in a sensor onboard the vehicle, logged data (before and after the fault) about the vehicle performance and the diagnostics (i.e. the observer-based state estimates and residuals) are sent to the RDC. At the RDC, the first and foremost task is to find out whether the fault is indeed a sensor fault. The other possibility is that wear-and-tear on the vehicle has led to

changed parameters in the car model (e.g. suspension stiffness, cornering coefficients). This possibility is verified or ruled out by simulating the existing model with the logged inputs and comparing the outputs of this simulation to the logged outputs. If there is indeed a change in vehicle parameters, system identification is performed using the structure of the original model and the logged data to get a new plant model for that particular vehicle. This model is then used to generate new controller and diagnostics code for that vehicle. Irrespective of whether there is a sensor fault or a change in vehicle parameters, a new optimal controller is formulated and a performance profile for that controller is generated. This new control controller is coded and sent to the vehicle, along with the performance profile for run-time verification of the new controller.

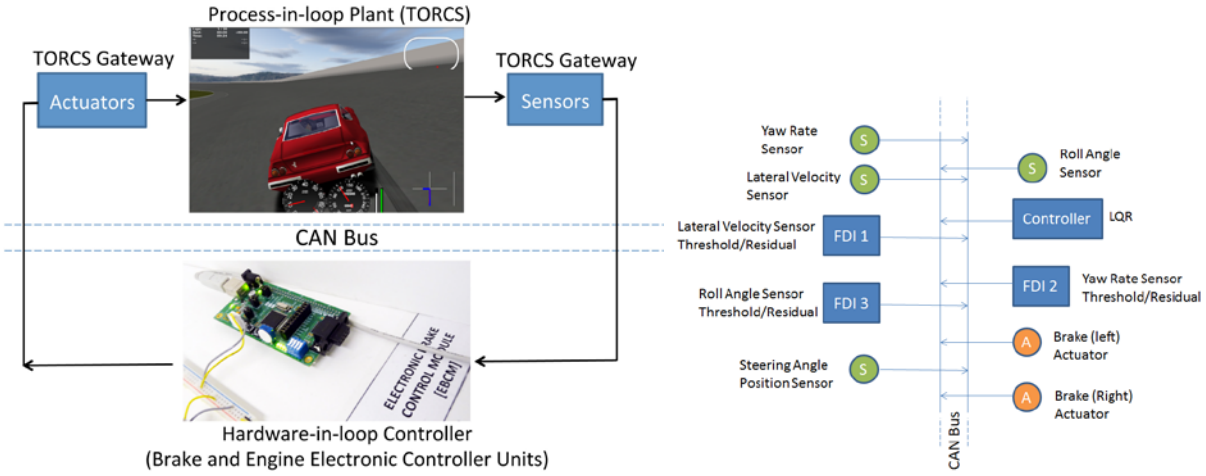


Figure 4. AutoPlug Test-bed Architecture with hardware-in-loop controllers

B. AutoPlug Testbed

The AutoPlug testbed is a Hardware-In-The-Loop simulation platform for ECU development and testing. The hardware is in the form of a network of ECUs, on which we implement the control and diagnostic algorithms. Each ECU runs the nano-RK RTOS, a resource kernel with preemptive priority-based real-time scheduling. Instead of a real-vehicle, our plant uses The Open-source Race Car Simulator (TORCS). This provides physics-based high-fidelity vehicle models and different road terrains. The testbed provides us with the realism of using a real vehicle, and also has enough flexibility to implement our own code. In addition, we can introduce faults that are not covered by set of standard Diagnostic Trouble Codes (DTC). We have tested out basic control algorithms, running as real-time tasks on nano-RK, for Anti-Lock Braking System (ABS), Traction Control, Cruise Control and Stability Control to see that the testbed indeed performs like a real vehicle would. AutoPlug is free and open-sourced.

The AutoPlug testbed consists of three layers, *Vehicle Dynamics Simulation*, *ECU Network* and the *middleware* for control algorithms, runtime software/system diagnosis, code updates and verification. The simulation layer models the dynamics of a vehicle (e.g. Toyota Corolla) on the physics-based simulator (TORCS). The ECU network consists of four embedded controllers (Freescale HCS12) networked over an industry standard CAN bus. The middleware executes on a small computer that provides a gateway protocol for vehicle manufacturers to interface with the ECU network and provides us with the simulated capabilities of a RDC. Fig. 4 shows a simplified view of the AutoPlug architecture.

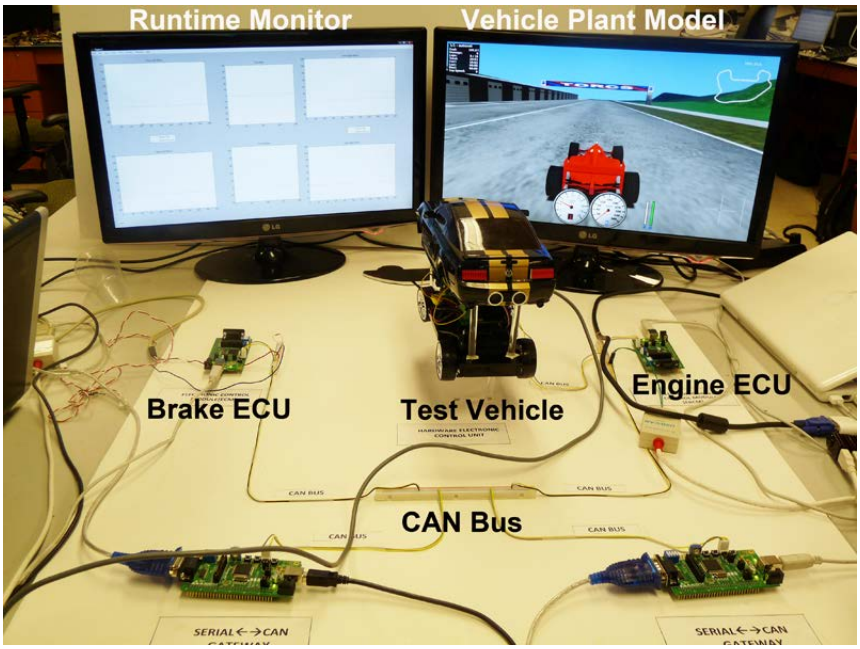


Figure 5. AutoPlug Test-bed with Engine and Brake ECUs running traction control, stability control, anti-lock braking, and adaptive cruise control

The four ECUs on the testbed are:

1) ***TORCS Gateway ECU:*** The simulation data (sensor values) are sent from TORCS (running on a computer) to the hardware-based ECUs in real-time over the CAN bus via this gateway. The inputs to the simulation in TORCS (e.g. steering angle, acceleration and brakes), from the ECUs and the user, are also received through the TORCS gateway and sent to the simulator.
2) ***Brakes ECU:*** This ECU controls the inputs to the brakes of the vehicle in TORCS. Based on the user inputs (which are sent over the CAN bus) and sensor values, the control algorithm on the ECU calculates the brake force for the individual wheels of the vehicle (generalized as the left or the right side of the vehicle). This is sent as a message over the CAN bus to the TORCS Gateway which in turn sends the calculated inputs to the vehicle actuator in TORCS.
3) ***Engine ECU:*** This ECU controls the acceleration, gear and clutch inputs to the vehicle. Algorithms like Cruise Control and Traction control are implemented onboard this ECU and it's tasked with modulating the user inputs to the vehicle and sending the modulated inputs as a message to the TORCS Gateway over the CAN bus.
4) ***MATLAB Gateway:*** In order to log data from the testbed and to monitor the vehicle parameters (like yaw rate etc.) in real-time, the MATLAB gateway reads the TORCS outputs, user inputs, and controlled inputs from the Brakes and Engine ECU from the CAN bus and sends them to the middleware. The middleware can also communicate with the other ECUs over the CAN bus if needed.

C. Code updates

Reprogramming the ECUs over the CAN bus was one of the features of the first generation AutoPlug framework. A smartphone can connect to the vehicle's CAN bus through the Onboard Diagnostics (OBD) port and the code update can be downloaded onto the smartphone. This downloaded code can then programmed onto the ECU over the CAN bus. Once a fault is detected onboard the vehicle, the Remote Diagnostics Center receives information of that fault, and new code is formulated at the RDC. The vehicle user then has a choice to initiate a code update remotely. The downloaded code is loaded in a secure manner from the owner’s smartphone into the vehicle via a WiFi gateway interfacing the vehicle’s on-board diagnostics (OBD) computer.

In the past year, we have extended AutoPlug to include Adaptive Cruise Control (ACC) – (for details see http://autoplug.blogspot.com/) to evaluate control algorithms and security attacks on ECUs. We have been able to successfully demonstrate the effect of sensor noise, limited field of view and other non-idealities on the performance of ACC.

C. Results

We conducted two case studies with AutoPlug. One with Electronic Stability Control (ESP), and the other for Diesel After-treatment related recalls. For the former, we applied system identification on the vehicle plant model to extract the lateral dynamics. The results re seen below:

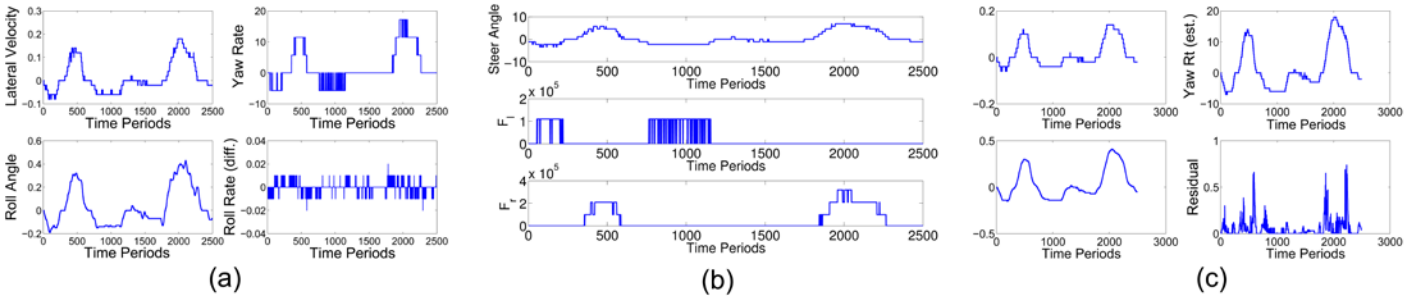


Figure 6. (a) Vehicle Lateral Velocity outputs for closed-loop operation with hardware ECUs; (b) Forces and steering input into the system. Note, the brake forces are regulated by the feedback controller, while the steering input is from the vehicle driver; (c) Estimates from the Kalman filter driven by sensor inputs

Faults were then injected as timing jitter, timing offsets and sensor noise. By applying the remote diagnostics procedures, we are able to derive adaptive fault residual thresholds to determine if the fault is with sensors or timing and not with normal driving behavior.

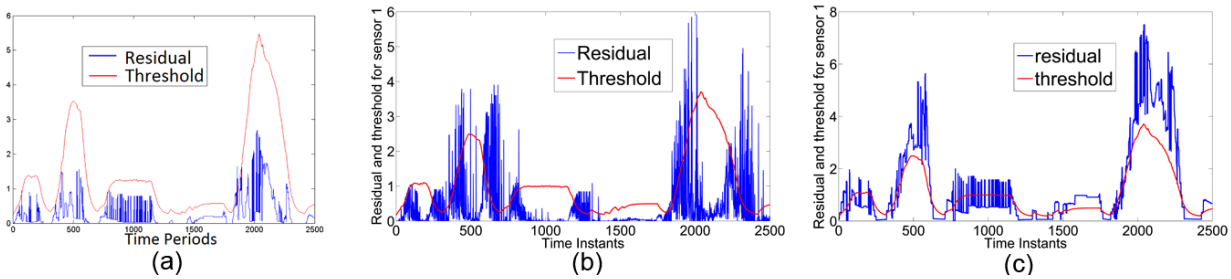


Figure 7. (a) Residual and thresholds for lateral velocity sensor, with fault in roll angle sensor; (b) Residual and threshold for jittery Lateral Velocity sensor; (c) Residual and threshold for lateral velocity sensor (calibration fault)

Conclusions

In the project, we introduce an architecture for merging on-board and remote diagnostics for automotive control systems that can potentially make vehicle recalls a less reactive process. We also show a diagnostics scheme for a feedback control system and evaluate it on the AutoPlug test-bed with reasonably good results. The overall scheme is relatively new, and potentially risky, but it can initially target non-critical control system, e.g. the vehicle body control systems. One particular example, which shows that this scheme could be useful, is the 936,000 vehicles that Honda had to recall due to issues with the power window and the automatic transmission. Other possible issues may arise with the safety of the Firmware-Over-The-Air (FOTA) approach for safety critical ECUs, where parties with malicious intent may hack into the network and compromise the safety of the vehicle by reprogramming the ECUs. So far, this has not been a part of our study, but Koscher et al. have extensively studied the security of modern automobiles. Also, security of Cyber Physical Systems (CPS) is of growing interest. A logical extension of our work is to focus on CPS security for networked automotive control systems, and is currently being looked at within Penn. A more

specific extension is to formulate a method to tune the design parameters of the threshold function in diagnostics scheme, which currently are experimentally chosen.

Recommendations developed as a result of the project:

AutoPlug is being applied to both software-based control system faults and also diesel after-treatment faults. The use of remote diagnostics will help reduce the human and economic cost of software-related vehicle safety recalls.